Vysoká škola báňská – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Generátor elektronických knih s náhledem knihy pro platformu Android

E-book Generator with Preview for Android Platform

2014 **Jaroslav Surala**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra informatiky

Zadání bakalářské práce

Student: **Jaroslav Surala**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R025 Informatika a výpočetní technika

Téma: Generátor elektronických knih s náhledem knihy pro platformu Android
E-book Generator with Preview for Android Platform

Zásady pro vypracování:

Využití mobilního telefonu nebo tabletu jako čtečky elektronických knih je jednou z oblastí, pro něž jsou tato zařízení často využívána. Z hlediska uživatelského komfortu je ovšem problémem nutnost použití PC k převodu (statických) WWW stránek (zejména mobilního webu) nebo existujícího obsahu do formátu, podporovaného většinou bezplatných čteček elektronických knih. Cílem této práce je vytvořit konvertor pro mobilní zařízení, který provede konverzi do alespoň dvou nejběžnějších formátů elektronických knih (nepoužívajícími DRM) a prostého textu z WWW stránek ve formátu HTML a RSS kanálů (včetně jejich případného stažení resp. načtení z paměťové karty). Součástí aplikace bude i možnost jednoduchého náhledu výsledné knihy pro účely kontroly převodu.

1. Prostudujte a stručně popište nejznámější formáty elektronických knih z hlediska možností formátování textu, hypertextových odkazů, reprezentace dat, snadnosti implementace a dostupnosti čteček těchto formátů na platformě Android.
2. Zvolte verzi Android OS/API, ve které budete konvertor realizovat, a popište případné nestandardní knihovny, které budete využívat.
3. Vyberte alespoň 2 vhodné formáty elektronických knih - společně s importem z HTML/RSS (popř. dalších vhodných formátů) a exportem do prostého textu - a implementujte konvertor s možností jednoduchého zobrazení náhledu vytvořeného výstupu.
4. Výsledné řešení otestujte alespoň na dvou různých mobilních zařízeních a zhodnoťte funkčnost a rychlost konverze.

Seznam doporučené odborné literatury:

[1] Burnette, E. Hello, Android: Introducing Google's Mobile Development Platform. Pragmatic Bookshelf, 2008. ISBN: 978-1-93435-617-3
[2] Meier, R. Professional Android 2 Application Development. Wrox Press Ltd., 2010. ISBN: 978-0-47056-552-0
[3] Konvertor Calibre [online]. 2011 [cit. 2011-09-20]. Dostupné z WWW: <http://calibre-ebook.com>
[4] World Wide Web Consorcium (W3C) [online]. [cit. 2013-09-20]. Dostupné z WWW: <http://www.w3.org>

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **Ing. Pavel Moravec, Ph.D.**

Datum zadání: 01.09.2013

Datum odevzdání: 07.05.2014

doc. Dr. Ing. Eduard Sojka
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

V Ostravě dne: *7. května 2014*

Svrala
.....
podpis studenta

Poděkování

Rád bych poděkoval Ing. Pavlu Moravcovi, Ph.D. za jeho cenné rady a čas, který si na mě udělal při tvorbě této bakalářské práce. Dále bych chtěl poděkovat Lucii Mlčůchové za vypůjčení mobilního telefonu, na kterém jsem testoval výslednou aplikaci.

Abstrakt

Práce se zabývá vývojem aplikace pro mobilní telefony a tablety s operačním systémem Android ve verzi 4.0 a vyšší. Aplikace umí stáhnout a uložit webové stránky, a poté realizuje konverzi do elektronické knihy nebo prostého textu. Dále aplikace umí stáhnout a zpracovat RSS kanál a nabízí možnost stažení jednotlivých článků. Další funkcí je zobrazení knih ve svém vlastním zobrazovači, nebo přímo z aplikace je rovněž možné otevřít knihu z externí čtečky, která podporuje daný formát. Práce také popisuje vybrané formáty, do kterých je realizována konverze.

Klíčová slova

Android, konverze, ebook, offline obsah, EPUB, FictionBook, HTML stránka, RSS kanál

Abstract

The thesis deals with development of an application for mobile devices and tables with Android operating system 4.0 and higher. The application can download web pages, save them and convert the content into ebooks or plain text files. Next the application can download and process RSS channels and offers the possibility to download individual articles. The application may display the generated ebook in its own preview or open the book with external applications that support the supplied ebook format. Individual formats selected for conversion implementation are also described.

Key words

Android, conversion, ebook, offline content, EPUB, FictonBook, HTML page, RSS channel

Seznam použitých zkratek

API – Application Programming Interface

CSS – Cascading Style Sheets

DOM – Document Object Model

ebook – elektronická kniha

HTML – Hypertext Markup Language

RSS – Really Simple Syndication

SDK – Software Development Kit

UI – User Interface

WHATWG – Web Hypertext Application Technology Working Group

XHTML – Extensible Hypertext Markup Language

XML – Extensible Markup Language

Obsah

Seznam obrázků

1 Úvod

Využití mobilního telefonu nebo tabletu se již v dnešní době neomezuje pouze na volání, posílání SMS zpráv či hraní her. Tato zařízení jsou často lidmi využívána jako čtečky elektronických knih. Ovšem z hlediska komfortu uživatele je problémem nutnost použití počítače k převodu webových stránek do formátu podporovaného většinou bezplatných čteček elektronických knih. Někteří uživatelé mobilního telefonu nemají Internet v mobilu (poskytovaný mobilními operátory), pro přístup k Internetu tedy využívají volně přístupné WiFi sítě, ty ovšem nejsou všude dostupné a uživatelé proto nemohou být stále online.

Cílem bakalářské práce je vytvořit aplikaci pro platformu Android, která bude provádět konverzi do alespoň dvou formátů elektronických knih a prostého textu z webových stránek. Aplikace bude stahovat webové stránky a provádět konverzi do formátu elektronické knihy nebo do prostého textu. Dále bude aplikace umět stahovat RSS kanál a zobrazovat jednotlivé články, pokud je uživatel připojen k Internetu nebo nabídne stažení jednotlivých článků. Aplikace bude nabízet vlastní jednoduchý zobrazovač pro vytvořenou knihu. Uživatel bude mít možnost přímo z aplikace vytvořenou knihu otevřít v jiné nainstalované aplikaci v systému, která podporuje daný formát elektronické knihy. Aplikace je určena pro lidi, kteří mají omezený přístup k Internetu — jednoduše si stáhnou stránky, které je zajímají a v době kdy nebudou online, si mohou jednotlivé stránky přečíst.

V druhé kapitole bude představen pojem ebook a formáty, které by měla aplikace vytvářet. V kapitole se budu zabývat vlastnostmi jednotlivých formátů a jejich odlišnostmi. Další částí bude porovnání některých dostupných elektronických čteček.

Ve třetí kapitole budou představeny některé technologie, které budou použity při vývoji aplikace. Představím některé části platformy Android, které budu využívat v aplikaci. Ve čtvrté kapitole popíši detailněji požadavky na aplikaci a její jednotlivé části.

Od kapitoly číslo pět až po kapitolu číslo sedm se budu zabývat samotným vývojem aplikace. Nejprve bude nastíněna analýza a návrh jednotlivých komponent aplikace. Dále budou ukázány některé vybrané procesy a okomentovány, bude zde i návrh uživatelského rozhraní. V poslední části se budu zabývat implementací aplikace. Budou popsány některé vybrané části, které okomentuji. Tato část bude obsahovat úryvky kódu.

V předposlední kapitole se budu věnovat testování aplikace na různých zařízeních. V testování se budu zaměřovat na rychlost konverze do jednotlivých formátů elektronické knihy. V závěru shrnu dosažené výsledky a zmíním, jak hodlám aplikaci v budoucnu rozšiřovat.

2 Elektronická kniha

Pojem ebook [1] (elektronická kniha) se nejčastěji používá pro označení digitálního ekvivalentu tištěné knihy. Někdy se pojem ebook označuje zařízení, které se specializuje na čtení takových souborů. Jako ebook bývají označovány různé formáty (např. EPUB, PDF, FB2 aj.).

Výhody

- Šetří místo
- Díky podsvícení lze číst i při slabém světle
- S ebook readerem se manipuluje snáze než s tištěnou knihou
- Knihy nejsou nikdy vyprodány
- Knihy mohou být poskytovány i zcela zdarma - bez nákladů na tisk, pokud se tak autor rozhodne

Nevýhody

- E-kniha vyžaduje ke čtení další zařízení. (Většinou to bývá specializovaná čtečka, aplikace pro mobil, tablet nebo počítač.)
- Zobrazení každého ebooku se musí přizpůsobit displeji zařízení
- Příliš mnoho formátů ebooků, každý s jejich vlastní čtečkou

2.1 EPUB

EPUB [2][3] je svobodný formát vytvořený pro ebooky. Je vytvořen dle standardu organizace International Digital Publishing Forum (IDPF). Aktuální je verze 3.0 schválená 11. října 2011. Přípona souboru je epub ačkoli se jedná o přejmenovaný zip soubor se specifickou adresářovou strukturou. Zip jako kontejner byl zvolen pro své rozšíření a známou specifikaci. Obsah je zapsán pomocí XML a HTML5 tagů. Splňuje tzv. reflowable, to znamená, že čtenář si může přizpůsobit rozvržení textu pro příslušné zařízení. Stal se neoficiálním českým formátem pro čtecí zařízení.

2.1.1 Vlastnosti EPUB formátu

Následuje výčet důležitých vlastností formátu. Ve výčtu nejsou uvedeny všechny vlastnosti formátu.

- Volný a otevřený formát
- Reflowable (zalamování) a změna velikosti textu
- Umožňuje různá zobrazení v závislosti na přístroji
- Podpora HTML5 vybraných tagů
- Rastrové a vektorové obrázky přímo v souboru publikace
- Podpora CSS

2.1.2 Adresářová struktura

Ve výpisu 1 je ukázka možné adresářové struktury formátu. Tuto adresářovou strukturu využiji při vytváření souboru ve své aplikaci. Soubor pojmenovaný mimetype je textový soubor bez přípony, který obsahuje pouze, deklaraci MIME (application/epub+zip). Je pro všechny knihy ve formátu EPUB jednotný. Ve složce *OEBPS* se nachází stránky, které budou obsaženy v knize. Dále soubor, který obsahuje odkazy na jednotlivé stránky (content). A také soubor, který vytváří obsah knihy a odkazuje na jednotlivé části knihy (toc). Ve složce *META-INF* se nachází soubor (container), který obsahuje seznam všech souborů obsahující odkazy na jednotlivé stránky. Ve složce *images* jsou uloženy jednotlivé obrázky. Pro správné zobrazení obrázků je potřeba ve stránce přepsat cestu k obrázkům.

```
mimetype
META-INF/
        container.xml
OEBPS/
        content.opf
        toc.ncx
        kniha, díl, kapitola.html
images/
        soubory lze i nořit.jpg
css/
        style.css
```

Výpis 1: Adresářová struktura [3]

2.1.3 Container soubor

Container slouží jako seznam všech souborů s příponou opf v knize. Zápis je pomocí XML syntaxe. Na obrázku 1 je zobrazen kód souboru. V elementu rootfile je uvedená cesta k souboru s příponou opf.

```
<?xml version="1.0"?>
<container version="1.0" xmlns="urn:oasis:names:tc:opendocument:xmlns:container">
    <rootfiles>
        <rootfile full-path="OEBPS/content.opf"
            media-type="application/oebps-package+xml" />
        <rootfile full-path="OEBPS/další OPF.opf"
            media-type="application/oebps-package+xml" />
    </rootfiles>
</container>
```

Obrázek 1 *Ukázka kódu souboru container*

2.1.4 Toc soubor

Toc je soubor sloužící pro orientaci v publikaci. Soubor určuje obsah knihy a řazení kapitol. Soubor je napsaný v XML syntaxi pomocí hypertextového odkazu. Ty se dělí na dva typy. Odkazuje se buď na celý soubor s příponou html (xhtml), viz obrázek 2. Ovšem také se může odkazovat na jeho jednotlivé části. Pokud se odkazuje na části textu v HTML souboru, musí být v souboru nastavena na příslušném místě kotva, která bude odpovídat odkazu v souboru s příponou toc. Obrázek 3 ukazuje odkazování pouze na část v jedné HTML stránce.

```
<navPoint id="cokoliv" playOrder="pořadové číslo">
    <navLabel><text>zobrazovaný název</text></navLabel>
    <content src="přílušný soubor.html" />
</navPoint>
```

Obrázek 2 *Ukázka kódu pro řazení jednotlivých souborů*

```
<navPoint id="cokoliv" playOrder="pořadové číslo">
    <navLabel><text>zobrazovaný název</text></navLabel>
    <content src="přílušný soubor.html#značka např.D1-CH3" />
</navPoint>
```

Obrázek 3 *Ukázka kódu pro řazení uvnitř souboru*

Na obrázku 4 je zachycen kompletní kód souboru toc. Kód naznačuje, jak se provádí jednotlivé řazení kapitol.

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE ncx PUBLIC "-//NISO//DTD ncx 2005-1//EN"
  "http://www.daisy.org/z3986/2005/ncx-2005-1.dtd">
<ncx xmlns="http://www.daisy.org/z3986/2005/ncx/"
  version="2005-1" xml:lang="en-US">
<head>
  <meta name="dtb:uid" content="uid"/>
  <meta name="dtb:depth" content="1"/>
  <meta name="dtb:totalPageCount" content="0"/>
  <meta name="dtb:maxPageNumber" content="0"/>
</head>
<docTitle><text>NÁZEV KNIHY</text></docTitle>
<docAuthor><text>autor</text></docAuthor>
<navMap>
  <navPoint id="titulní_strana" playOrder="1">
    <navLabel><text>název knihy</text></navLabel>
    <content src="book.html#start"/>
  </navPoint>
  <navPoint id="předmluva" playOrder="2">
  <navLabel><text>předmluva</text></navLabel>
  <content src="book.html#preface"/>
  </navPoint>
  <navPoint id="obsah" playOrder="3">
    <navLabel><text>Obsah</text></navLabel>
    <content src="obsah.html"/>
  </navPoint>
  <navPoint id="navpoint-1" playOrder="4">
    <navLabel><text>Kapitola 1:aneb jak to začalo</text></navLabel>
    <content src="book.html#S1"/>
  </navPoint>
  <navPoint id="navpoint-2" playOrder="5">
    <navLabel><text>Kapitola 2</text></navLabel>
    <content src="book.html#S2"/>
  </navPoint>
    <navPoint id="navpoint-3" playOrder="6">
    <navLabel><text>Kapitola 3: aneb jak to dopadlo</text></navLabel>
    <content src="book.html#S3"/>
  </navPoint>
</navMap>
</ncx>
```

Obrázek 4 *Kód souboru toc*

2.1.5 Konverze HTML stránky do EPUB formátu

EPUB je založený na XML, potřebujeme tedy vytvořit sadu XML souborů, zazipovat je do jednoho archívu. Výslednému archívu změníme koncovku na epub [5].

0. **Vytvoření HTML –** kniha je psaná v HTML s CSS styly, je potřeba dodržet formát XHTML. Tudíž pokud není soubor psaný již v XHTML (uzavírání elementů, používání uvozovek kolem atributů, atd..) potřebujeme soubor opravit do XHTML podoby. Pokud použijeme pro každou kapitolu samostatné XHTML soubory, umístíme všechny do stejné složky.
1. **Vytvoření MIME souboru –** v textovém editoru napíšeme application/epub+zip a uložíme bez přípony. Dáme do složky s XHML soubory.
2. **Přidání CSS stylu –** můžeme vytvořit dva druhy stylu pro knihu. Jeden pro stránky a jeden pro styl celé knihy. CSS soubory uložíme do složky k XHTML souborům nebo do jiné složky. Potom je ale nutné zpravit ve stránce odkaz na CSS soubor.
3. **Vytvoření titulní strany –** nemusíme používat obálku jako titulní stránku ale ve většině případů se to dělá. Pro přidání titulní strany vytvoříme XHTML soubor.
4. **Vytvoření obsahu –** vytvoříme soubor toc s příponou ncx. Je to XML soubor a měl by odkazovat na všechny XHTML soubory v knize.
5. **Vytvoření container XML souboru -** vytvoříme soubor, který se bude nazývat container a uložíme ho do adresáře *META-INF*. Struktura souboru je uvedena v kapitole 2.1.3.
6. **Vytvoření seznamu obsahu –** vytvoříme soubor content s příponou opf. Jedná se o soubor, který popisuje knihu. Soubor zahrnuje informace o knize (autor knihy, datum publikování, pohlaví autora, odkazy na jednotlivé HTML soubory).
7. **Závěr tvorby –** pokud máme vytvořeny všechny potřebné soubory, měli by být v jednom adresáři (kromě souboru container, který je uložen v *META-INF*). Jakmile si soubor pojmenujeme, vše zazipujeme do formátu zip. Poté už jen přejmenujeme, archív s příponou zip na soubor s příponou epub a elektronická kniha je vytvořena.

2.2 FictionBook

FictionBook [4] je otevřený ebook formát založený na jazyce XML. Soubor má příponu fb2. FictionBook formát neurčuje vzhled dokumentu, ale strukturu dokumentu. Všechna metadata, jako jméno a příjmení autora, název, datum vydání a podobně jsou také obsažena v souboru. Formát je vhodný pro automatické zpracování. Umožnuje automatickou konverzi do jiných formátů. Tvorba ebooku se řídí XML schématem, které specifikuje strukturu souboru.

Vlastnosti

- Zdarma a otevřený formát
- Podporuje reflow
- Jednoduché sémantické značkování
- Podporuje unicode
- Dokument může obsahovat
  - Strukturovaný text organizovaný ve vnořených sekcích
  - Citace
  - Odkazy a poznámky pod čarou
  - Rastrové obrázky (JPEG, PNG)
  - Tabulky (nepodporují je všechny čtečky)
- Inline formátování
  - Tučné (obvykle bold)
  - Zvýrazněné (obvykle italic)
  - Přeškrtnuté
  - Nadpis
  - Programový kód (obvykle v monospace font)

Rozdíl oproti jiným formátům

V protikladu s ostatními formáty (např. EPUB), se FictionBook skládá pouze z jednoho XML souboru. Obrázky jsou konvertovány do base64[1] a umístěny uvnitř tagu binary, takže velikost vložených obrázků se zvýší. Často je distribuován v zip archivu a hardwarové i softwarové čtečky jej umí číst přímo. Metadata a čistý text jsou vložena na začátek souboru, zatímco obrázky na konec souboru. To umožňuje softwaru vykreslit nebo zpracovat FictionBook než jsou soubory zcela přístupné.

[1] Base64 – schéma patřící do skupiny „binary-to-text“ kódování, které reprezentuje binární data pomocí tisknutelných znaků ASCII.

2.3 Dostupné aplikace

Pro platformu Andorid existuje řada čteček elektronických knih, které naleznete na Google Play. Z těch známějších a kvalitních je to například Moon+ Reader, který je ke stažení zdarma. Aplikace je i v placené verzi, která obsahuje některé další možnosti. Dalšími kvalitními bezplatnými aplikacemi pro čtení elektronických knih jsou Cool Reader a FBReader. Na Google Play je celá řada dalších čteček. Zmínil jsem ty, které mají větší počet stažení a hodnocení.

Tabulka 1 *Porovnání aplikací*

	Moon+ Reader	**Cool Reader**	**FBReader**
EPUB	Ano	Ano	Ano
FB2	Ano	Ano	Ano
TXT	Ano	Ano	Ano
DOC (Microsoft Word)	Ne	Ano	Ano
Online katalog (OPDS)	Ano	Ano	Ano
Předčítání	Ano (v PRO verzi)	Ano	Neuvedeno
Automatické rolování	Ano	Ano	Neuvedeno
Podpora slovníků	Ano	Ano	Ano
Denní/noční režim	Ano	Ano	Ano
Možnost "Ochrany očí"	Ano	Ne	Ne
Hodnocení	4,5	4,6	4,6
Počet stažení	128 977	103 995	76 150

V tabulce 1 je vidět porovnání aplikací. V tabulce nejsou zachyceny všechny možnosti čteček. Jednotlivé čtečky zvládnou otevřít více formátů, než je uvedeno v tabulce, uvedl jsem pouze ty, které budu vytvářet a formát dokumentu vytvořený nástrojem Microsoft Word. Dále jsem se zaměřil na parametry, které mohou čtenáři usnadnit čtení knihy, například zda čtečka obsahuje denní a noční schéma. Z tabulky je patrné, že aplikace jsou si co do možností velice podobné.

3 Použité technologie

Tato kapitola obsahuje popis použité externí knihovny a Android API použité při implementaci této aplikace. V kapitole 3.1 popíši externí knihovnu Jsoup. V kapitole 3.2 metodu zpracování XML souboru. A od kapitoly 3.3 se budu věnovat Andorid API.

3.1 Knihovna Jsoup

Jsoup [6] je Java knihovna pro práci a zpracování HTML. Poskytuje API pro extrakci dat a manipulaci s daty. Zpracovává data pomocí DOM, CSS a jquery. Jsoup implementuje WHATWG HTML5 specifikaci HTML5 a zpracovává HTML do stejného DOM jako, to dělají moderní prohlížeče.

Jsoup dokáže zpracovávat více vstupů:

- **Řetězec –** nejjednodušší způsob předání dat k pársování, který obsahuje HTML kód
- **Soubor –** další způsob je předání souboru, který obsahuje HTML kód. Výhodou je, že se nemusíme starat o zavírání či otevírání souboru.
- **URL –** asi nejčastější způsob je předání URL stránky, kterou chceme zpracovávat. Kompletní zdrojový kód se načte do paměti. Využití DOM.

Pro nalézaní a extrahování dat používá průchod DOM nebo CSS selektory. Knihovna dokáže manipulovat s HTML elementy, atributy a textem. Dokáže také vyčistit HTML kód od tagů, které jsou nebezpečné, nebo je nechcete zahrnout do výstupu. Čistění provádí pomocí white listů. Výstupem je čisté a bezpečné HTML. Knihovna je navržena tak, aby se vypořádala se všemi variantami HTML, které se na internetu vyskytují. Jsoup je open source projekt distribuovaný pod liberální licencí MIT. Zdrojový kód knihovny je dostupný na GitHub (https://github.com/jhy/jsoup/).

Při vývoji aplikace budu využívat všech těchto možností knihovny. HTML kód budu pročišťovat přes mnou definovaný white list. A při práci se stránkou její kód načtu do paměti.

3.2 SAX parser

Simple API for XML [7] umožnuje sériové zpracování XML dokumentu. Je to jiná varianta zpracování dokumentu oproti DOM. Jedná se o proudové zpracování, při kterém se dokument rozdělí na části (počáteční a koncové značky, obsahy elementů, komentáře, atd.). Postupně se vyvolávají události a způsob zpracování těchto událostí je na programátorovi. Tato metoda se využívá při zpracování dokumentu, kdy nechceme náhodně přistupovat, ale chceme sekvenční přístup k dokumentu. Výhodou je, rychlost zpracování a menší paměťové nároky. Oproti DOMu bývá rychlejší. Používá se pro zpracování větších XML dokumentů.

Při vývoji aplikace budu používat SAX parser pro zpracování RSS kanálu, který je strukturován jako XML dokument. Výsledkem bude RSS kanál článků.

3.3 AsyncTask

AsyncTask [8] je abstraktní třída, která vytvoří asynchronní vlákno, kdy výpočet běží ve vlákně na pozadí a po dokončení výpočtu je předán výsledek hlavnímu vláknu. Třída se sama postará o vytvoření vlákna, není tedy nutná manipulace s vlákny. Asynchronní úkol je definován třemi generickými typy (Params, Progress, Result) a čtyřmi stavy (onPreExecute, doInBackground, onProgressUpdate, onPostExecute).

Hlavní stavy:

- **onPreExecute** – stav, který je vyvolán před začátkem provádění úkolu. Používá se pro nastavení úkolu, zobrazení progress baru v uživatelském rozhraní.
- **doInBackground(Params…)** – stav, který je vyvolán ihned po dokončení předchozího stavu. Tento stav je určen pro realizování úkolu na pozadí, který může trvat dlouho.
- **onProgressUpdate(Progress…)** – tato metoda je použita pro zobrazení pokroku v uživatelském rozhraní zatímco provádění úkolu pokračuje na pozadí.
- **onPostExecute(Result) –** tento stav je vyvolán po ukončení úkolu. Výsledek operace provedené na pozadí je do této metody vložen jako parametr.

Při vývoji aplikace budu využívat této třídy pro stahování stránek z internetu a jejich následné uložení.

3.4 Activity

Activity (aktivita) [9] je komponenta, která poskytuje obrazovku, se kterou uživatel může interagovat. Každá aktivita má své okno, do kterého vykreslí uživatelské rozhraní. Aplikace se většinou skládají z více než jedné aktivity, které jsou k sobě volně vázány. Typicky je jedna aktivita označena jako „main activity“, která se spustí po spuštění aplikace. Každá aktivita může být spuštěna z jiné aktivity, za účelem provedení nějaké akce. Pokaždé když se spustí nová aktivita, tak je předchozí aktivita pozastavena. Systém si ji uchová v zásobníku („back stack“). Když se spustí nová aktivita, uloží se do zásobníku. Zásobník je typu LIFO, takže když uživatel ukončí aktuální aktivitu (tlačítko zpět), vymaže se ze zásobníku a načte se předchozí aktivita.

Aktivita se může nacházet v různých stavech životního cyklu (vytvořena, spuštěna, pozastavena nebo zničena). Pokaždé když aktivita změní svůj stav, je vyvolána metoda zpětného volání (callback method). Každá tato metoda umožňuje provést nějakou operaci, která se hodí k tomuto stavu. Například při zastavení uvolnit velké objekty.

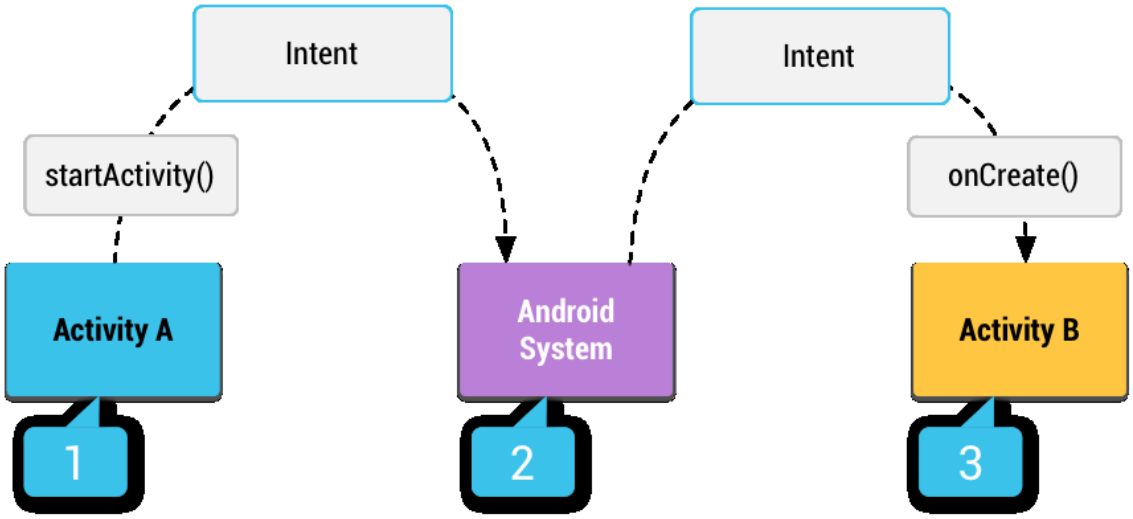
3.5 Intent

Intent [10] je zpráva, kterou můžeme požádat o akci z jiné komponenty. Intenty usnadňují komunikaci mezi komponentami různými způsoby. Existují tři základní typy.

- **Pro spuštění aktivity** – spuštění aktivity předáním intentu do metody **startActivity()**. Intent popisuje aktivitu, kterou má spustit a obsahuje veškerá nezbytná data. Chceme - li získat výsledek když aktivita skončí, použijeme metodu **startActivityForResult()**. Aktivita, která spouští jinou aktivitu obdrží výsledek jako samostatný intent v callback metodě **onActivityResult()**.
- **Pro spuštění služby** – služba je komponenta, která provádí operace na pozadí bez uživatelského rozhraní. Můžeme spustit službu pro provedení jednorázové operace (např. stáhnutí souboru) předáním intentu do metody **startService()**. Intent popisuje službu, kterou má spustit a obsahuje veškerá potřebná data
- **Pro doručení broadcastu** – broadcast je zpráva, kterou může přijmout každá aplikace. Systém poskytuje různé broadcasty pro systémové události (například zařízení se začalo nabíjet). Můžeme doručit broadcast do jiné aplikace pomocí předání intentu metodám **sendBroadcast(), sendOrderedBroadcast()**, nebo **sendStickyBroadcast()**.

Existují dva typy intentu:

- **Explictiní intent** – specifikuje komponentu kterou chce spusti pomocí jména (celé jméno třídy). Tento typ intentu typicky použijeme ve své vlastní aplikaci protože přesně známe název třídy aktivity nebo služby, kterou chceme spustit.
- **Implicitní intent** – nespouští komponentu pomocí jména, ale místo toho specifikuje akci, kterou chce provést. Například pokud chceme pořídit fotku, můžeme použít implictiní intent a požádat aplikaci, která je schopná pořídit fotku.



Obrázek 5 *Ukázka spuštění aktivity intentem – převzato z [10]*

3.6 Možnosti uložení dat

Systém Andorid umožnuje několik možností, jak uložit perzistentně aplikační data. Možnost, kterou si vyberete, záleží na specifických potřebách - jestli data jsou přístupná pouze Vaší aplikaci nebo jsou přístupná ostatním aplikacím (a uživateli). A také podle toho kolik vyžadují data místa. Možnosti uložení dat jsou následující [11].

- **Shared Preferences –** uloží primitivní datové typy (boolean, float, int, long, a string) v páru hodnota klíč.
- **Internal storage –** uloží data v paměti přístroje a jsou přístupná pouze aplikaci, která je uloží.
- **External storage –** uloží data na sdílené externí úložiště. Data jsou přístupná všem aplikacím i uživateli.
- **SQLite database –** uloží data strukturovaně v privátní databázi. Přistup má pouze aplikace.

V aplikaci budu používat external storage, aby mohli uživatelé aplikace přidávat soubory do adresářové struktury, manipulovat s nimi a soubory byly dostupné externím čtečkám.

4 Detailní popis aplikace

Tato kapitola se zabývá detailním popisem aplikace a jednotlivými částmi aplikace. Jsou zde rozvedeny požadavky na aplikaci.

Popis aplikace

Converter bude aplikace, která umožní stažení webové stránky z Internetu a převedení do fomátu elektronické knihy nebo prostého textu. Stahování bude umožněno jak na úrovni jedné stránky, tak na úrovni více stránek. Aplikace nabídne možnost stáhnutí stránky s obrázky nebo bez obrázků. Uživatel bude mít na výběr ze tří formátů, do kterých může stránky převést. Aplikace umožní stažení RSS kanálu a zobrazení článků, pokud bude přístup k síti. Bude zde i možnost pro stažení jednotlivých článků z kanálu. Aplikace bude mít svůj vlastní průzkumník pro možnost výběru knihy ke čtení. Rovněž bude obsahovat jednoduchý zobrazovač knih. Bude zde i možnost přímo z průzkumníku, otevřít knihu z nainstalované aplikace v systému (CoolReader, Moon+ Reader, apod.).

Získávání stránek

Získávání stránek bude možno provádět dvěma způsoby. První možnost je zadání odkazu na stránku přímo v aplikaci. A druhý způsob je využití možnosti prohlížeče a sdílet stránku aplikaci (v prohlížeči *sdílet stránku* a v nabídce výběr aplikace Converter). Stránky budou před uložením pročištěny od škodlivého kódu a částí, které nepotřebujeme. Bude navržen seznam tagů, které program vyfiltruje z původní stránky.

RSS kanály

Aplikace nabídne rovněž více možností zadání zdroje. Při spuštění zobrazí v rozbalovacím seznamu předdefinované zdroje, které boudou načteny ze souboru. Další možností bude opět sdílení odkazu z prohlížeče. Pokud bude mít aplikace přístup k internetu, může se chovat jako RSS čtečka, tedy jednotlivé články kanálu zobrazovat. Bude zde i možnost pro uživatele, kteří nebudou online po celou dobu a mohou si články stáhnout a převést do elektronické knihy.

Převod stránek

Při převodu stránek do formátu elektronické knihy bude kladen nárok, aby aplikace při provádění operace nezamrzla. Bude na výběr ze tří formátů (EPUB, FictonBook, textový soubor). Pro zobrazení knih bude implementován jednoduchý vlastní zobrazovač. Aplikace nabídne průzkumníka souborů a možnost otevření knihy z průzkumníka v nainstalované aplikaci nebo ve vlastním zobrazovači.

5 Analýza a návrh

Tato kapitola se zabývá analýzou a návrhem aplikace. Stručně popíši diagram návrhu tříd. Dále se budu věnovat vybraným procesům, které bude aplikace obsahovat a popíši je. V kapitole se také budu zabývat návrhem uživatelského rozhraní.

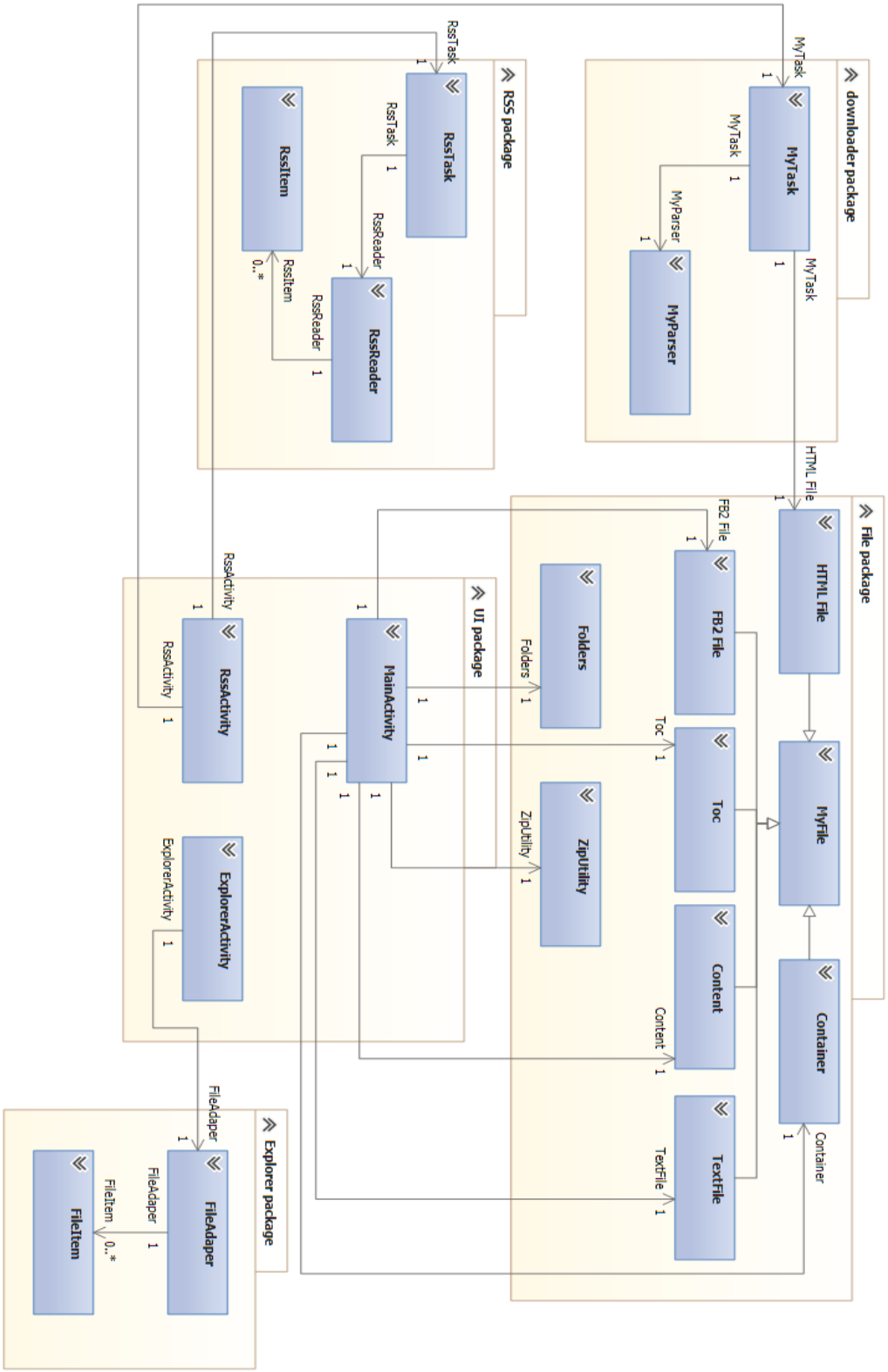
5.1 Diagram tříd

Na obrázku 6 je znázorněn diagram tříd, který popisuje jádro aplikace. Třídy jsou rozděleny do jednotlivých balíčků podle jejich funkcionality. Balíček *Downloader* obsahuje třídy, které se starají o stažení stránky. Balíček *File* se stará, o vytváření veškerých souborů, potřebných ke konverzi a také o vytvoření adresářové struktury na úložišti. Balíček *RSS* se stará o stažení a zpracování RSS kanálu. Balíček *Explorer* se stará o vytvoření adresářové struktury vnitřního uložiště mobilního telefonu. Balíček *UI* poskytuje třídy, které zprostředkovávají interakci s uživatelem.

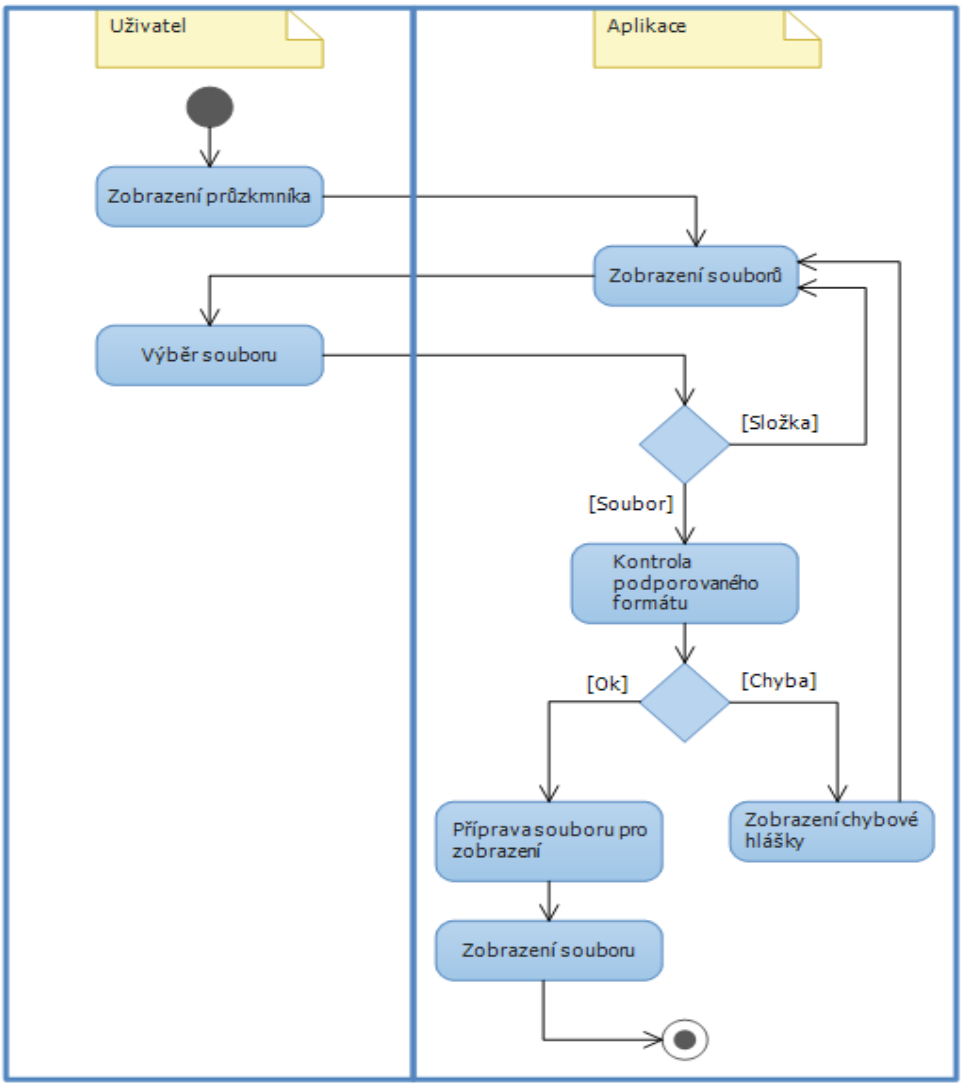
5.2 Diagramy aktivit

Na obrázku 7 je znázorněný diagram aktivit popisující výběr knihy pro přečtení. Akce začíná u uživatele, který spustí průzkumník. Aplikace zobrazí výpis souborů a složek z hlavního externího úložiště. Uživatel potom vybere soubor, který chce zobrazit nebo složku, ve které se soubor nachází. Pokud zvolí složku aplikace, zobrazí výpis souborů a složek této složky. Pokud zvolí soubor aplikace, zkontroluje zda, se jedná o podporovaný formát. V případě, že se nejedná o takový formát aplikace, zahlásí uživateli chybu a ten pokračuje dále ve výběru souboru. V případě, že se jedná o podporovaný formát, aplikace připraví soubor. Ve chvíli, kdy je soubor připravený aplikace zobrazí knihu uživateli.

Na obrázku 8 je znázorněný diagram aktivit popisující průběh stažení souboru. V diagramu se bere v potaz, že jsme korektně připojeni k Internetu, proto není kontrola stavu připojení zaznamenána v diagramu. Také je již vybrána stránka pro stahování. Akce začíná tedy uživatelem, který spustí stahování stránky. Aplikace následně zobrazí okénko s možnosti stahování (s obrázky, stahovat další stránky). Uživatel si zvolí možnosti, které požaduje a potvrdí. Aplikace vyhodnotí možnosti a na základě tohoto vyhodnocení pokračuje. Nejprve stáhne stránku. Pokud uživatel chtěl stránku i s obrázky, stáhne následně obrázky a uloží je. Poté stránku pročistí a uloží. Pokud uživatel nechtěl, stránku s obrázky stránku pouze pročistí a uloží. Jestliže uživatel vybral jako možnost stažení i dalších stránek, tak aplikace najde odkazy na stránce a postupně začne stahovat a ukládat stránky podle vybraných možností uživatele.



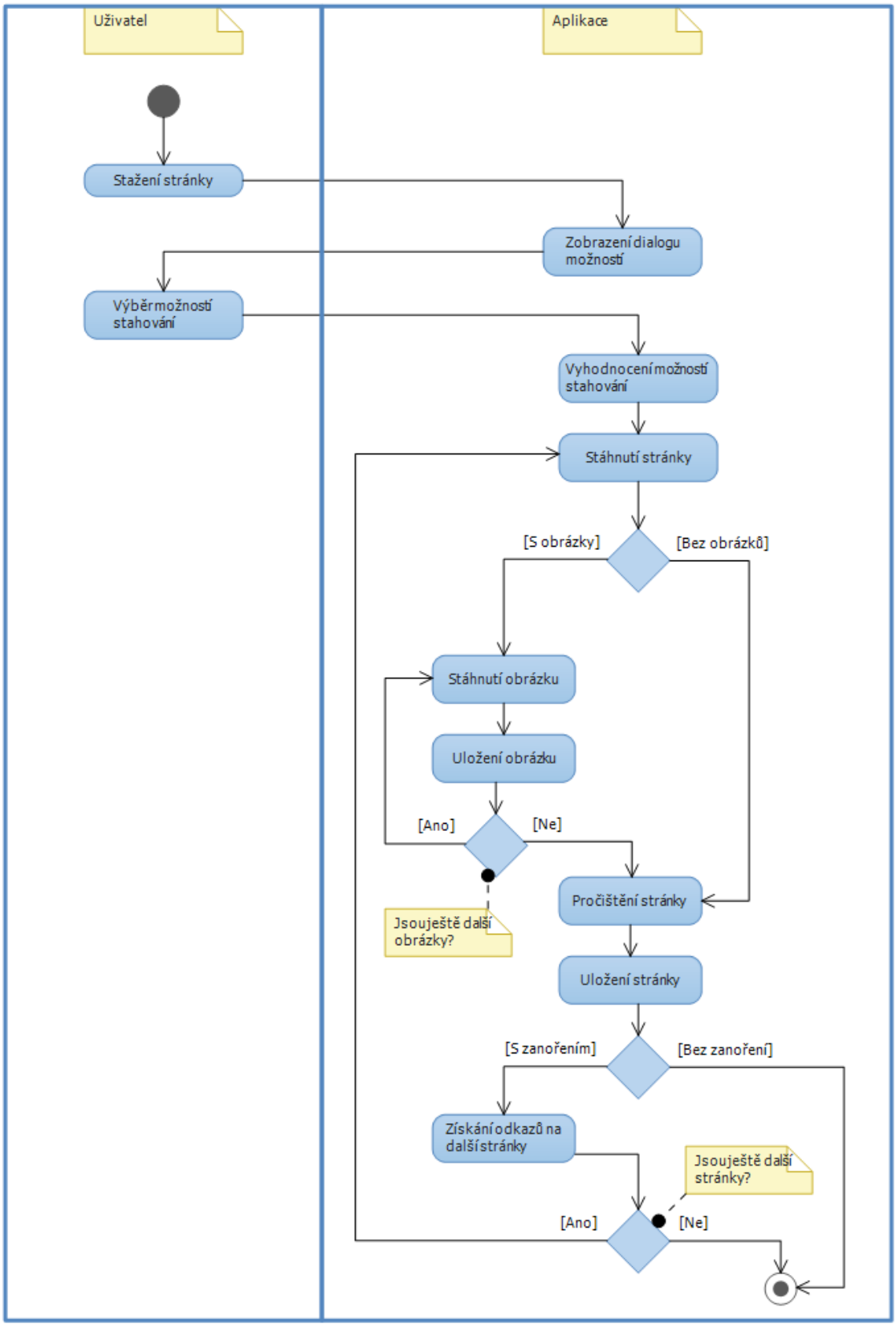
Obrázek 6 *Zjednodušený diagram tříd jádra aplikace*



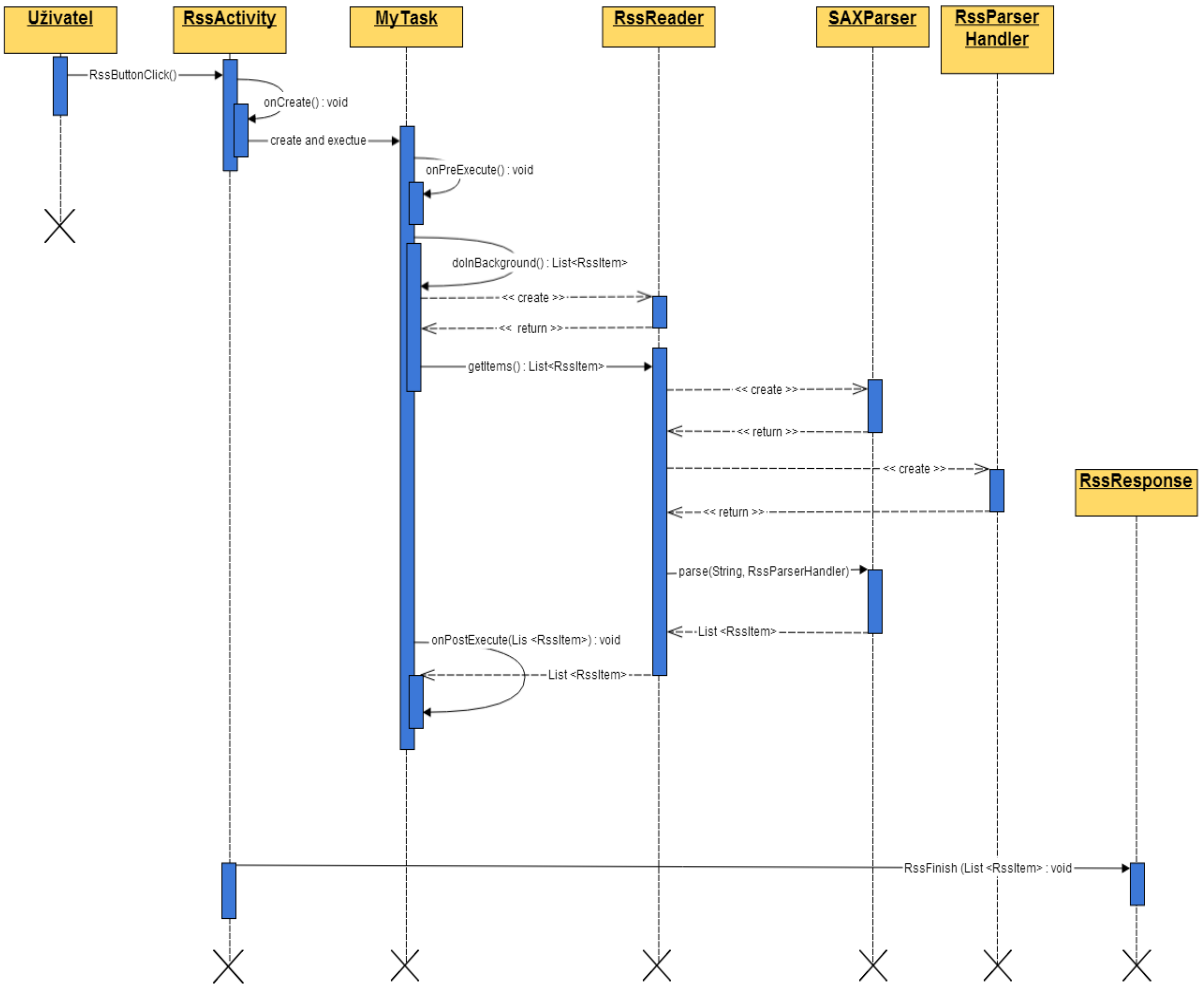
Obrázek 7 *Diagram aktivit – výběr knihy*

5.3 Sekvenční diagram

Na obrázku 9 je znázorněný sekvenční diagram pro stažení RSS kanálu. Sekvence začíná uživatelovým stiskem tlačítka pro stažení RSS kanálu. Spustí novou aktivitu *RssActivity*, která ve své metodě *onCreate()* spustí stahování RSS kanálu metodou *execute()* třídy *RssTask*. Následně v metodě *onPreExecute()* inicializuje vše potřebné pro průběh operace. Po dokončení inicializace je vyvolána metoda *doInBackground(String…)*, která začne provádět operaci metodou *getItems()* třídy *RssReader*. Metoda zpársuje kanál do jednotlivých položek. Po dokončení je vyvolána metoda *onPostExecute(List<RssItem>)*, která obdrží jako parametr seznam položek a zobrazí seznam uživateli.



Obrázek 8 *Diagram aktivit – stažení stránky*



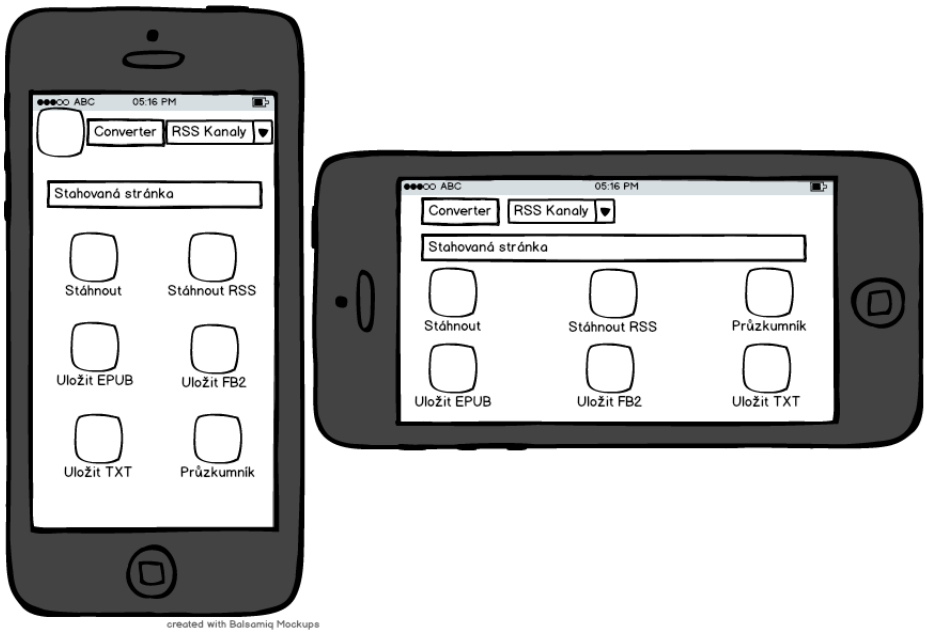
Obrázek 9 *Sekvenční diagram – stažení RSS kanálu*

5.4 Návrh UI

Tato kapitola se bude zabývat návrhem uživatelského rozhraní. Budou předvedeny jednotlivé aktivity (okna) aplikace, se kterými bude uživatel interagovat.

5.4.1 Hlavní menu

Zobrazí se při spuštění aplikace. Jsou zde umístěny tlačítka, pokrývající všechny základní operace. Hlavní menu je rozloženo a vytvořeno tak, aby se v něm vyznal i začínající uživatel aplikace. Je zde pole, ve kterém je zobrazen link na aktuální webovou stránku, se kterou aplikace bude pracovat. Tlačítko s obrázkem šipky pro stažení a uložení stránky. Dále jsou zde tři tlačítka pro převedení stránky do elektronické knihy. Každé z tlačítek má logo formátu, do kterého provede konverzi. Poslední tlačítko slouží pro spuštění průzkumníka. Na obrázku 10 je zachycen návrh uživatelského rozhraní hlavního menu.

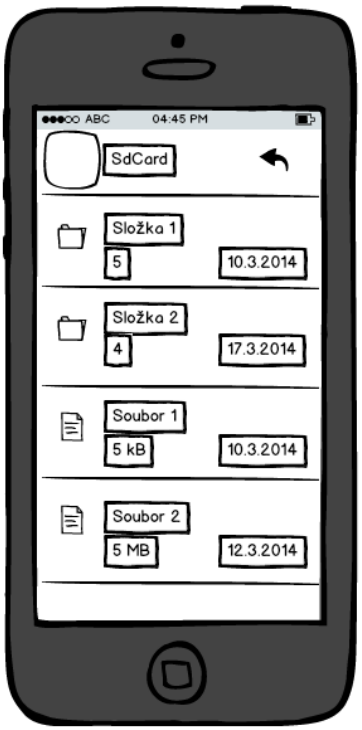


Obrázek 10 *UI hlavního menu*

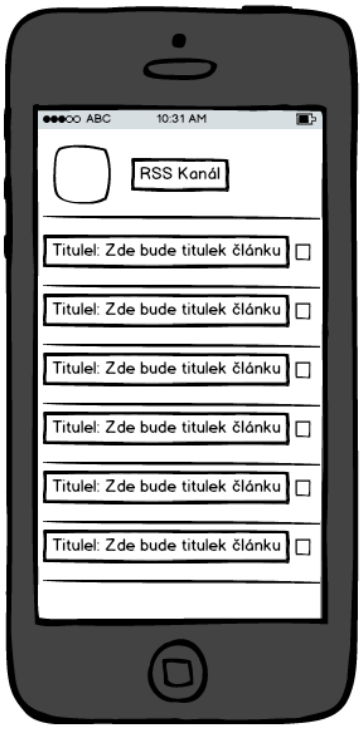
5.4.2 Průzkumník a RSS kanál

Aktivita (okno) pro průzkumník souborového systému se zobrazí po stisku tlačítka *Průzkumník* v hlavním menu. Při zobrazování souborového systému je kladen důraz na rozlišení složky od souboru a také jednotlivých druhů souborů navzájem. V hlavičce je zobrazen název aktuálního adresáře. A vedle názvu adresáře je tlačítko pro navrácení se v adresářové struktuře výše. Na obrázku 11 je zachycen návrh uživatelského rozhraní průzkumníku.

Aktivita (okno) pro zobrazení zpracovaného RSS kanálu se spustí, po stisku tlačítka *Stáhnout RSS* v hlavním menu. Zobrazí jednotlivé titulky článků a vedle nich zaškrtávací pole pro možnost výběru stránek pro stahování. Na obrázku 12 je zachycen návrh pro tuto aktivitu.



Obrázek 11 *UI průzkumníku*



Obrázek 12 *UI RSS kanálu*

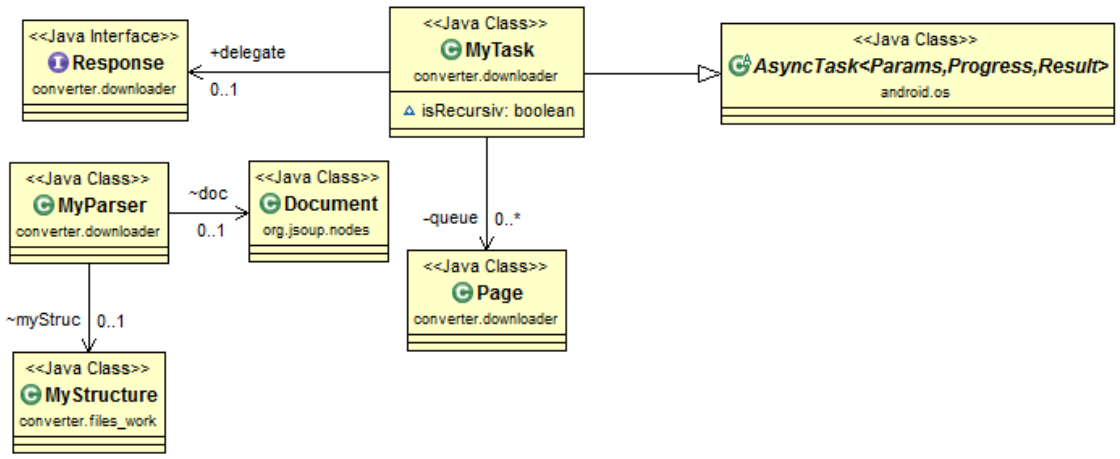
6 Implementace

V této kapitole se budu věnovat samotné implementaci aplikace. Popíši vývoj aplikace, a jek je strukturována. Ukáži vybrané části kódu a okomentuji je. K vývoji bylo použito vývojové prostředí Eclipse, do kterého bylo doplněno Android SDK. Nejnižší možná verze Androidu pro spuštění aplikace bude Andorid 4.0 (ICE CREAM SANDWICH) kvůli kompatibilitě některých částí. Navíc vlastním mobilní telefon s verzí Andorid 4.1.2 (JELLY BEAN) a tedy nejčastější testování bude probíhat na tomto zařízení.

6.1 Balíčky a diagramy tříd

Aplikace je logicky oddělena do balíčků. Každý balíček obsahuje třídy podle jejich funkcionality. V hlavním balíčku *conververt.main* se nachází třída, která se stará o zobrazení hlavního okna aplikace. A dále je zde třída, která slouží k zobrazení převedených knih pro formát FictionBook a textového souboru.

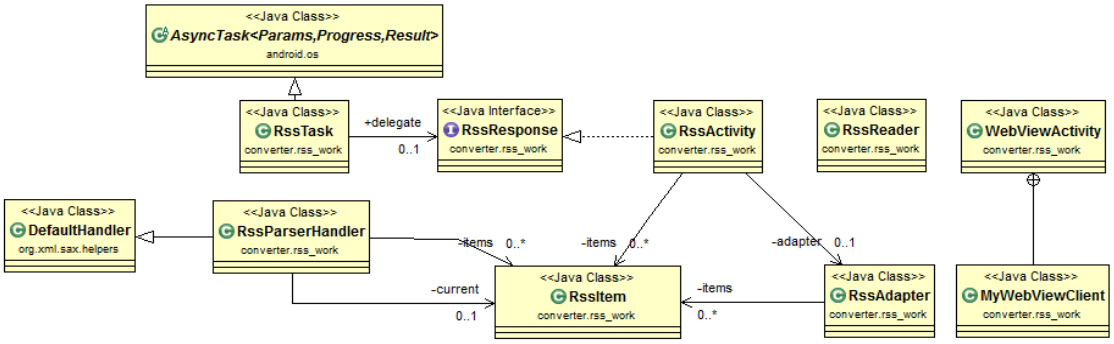
V balíčku *converter.downloader* se nachází třídy, které se starají o stažení a uložení stránky na externí úložiště. Nachází se zde i třídy, které se starají o pročištění stažené stránky. Obrázek 13 zachycuje diagram tříd balíčku. Stěžejní třídou balíčku je třída *MyTask*. Tato třída se stará o spuštění stahování v asynchronním vlákně a po dokončení stahování stránku pročistí a uloží. O pročistění stažené stránky se stará třída *MyParser*.



Obrázek 13 *Diagram tříd balíčku converter.downloader*

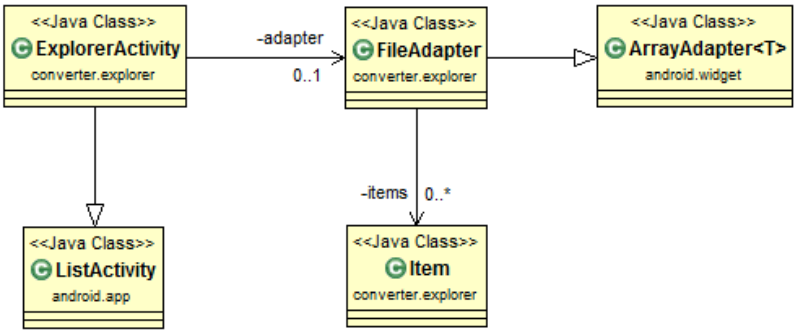
V balíčku *converter.files_work* se nachází třídy pro vytváření potřebných souborů. Je zde i třída, která se stará o vytvoření adresářové struktury na externím úložišti mobilního telefonu a abstraktní třída, která se stará o zápis na externí úložiště, z níž dědí všechny třídy, které se starají o vytváření souboru.

V balíčku *converter.rss_work* se nachází třídy, které se starají o stažení a zobrazení RSS kanálu. Obrázek 14 zachycuje diagram tříd balíčku. Základní třídou je *RssTask*, která se stará o vytvoření asynchronního vlákna a spuštění operace parsování RSS kanálu. Třída *RssReader* je třída starající se o parsování, které je implementováno pomocí SAX parseru. Dále je zde třída *RssActivity* pro zobrazení seznamu článku v kanále. Třída *WebViewActivity* se stará o zobrazení jednotlivých článků v případě připojení k internetu.



Obrázek 14 *Diagram tříd balíčku converter.rss_work*

V balíčku *converter.explorer* se nachází třídy pro vytvoření a práci s průzkumníkem. Obrázek 15 zachycuje diagram tříd balíčku. Hlavní třídou je *FileAdapter*, která se stará o vytvoření vzhledu jedné položky v seznamu. Dále třída *Item*, reprezentuje jednu položku seznamu a určuje vzhled položky na základě rozhodnutí, jestli se jedná o soubor nebo složku. Třída *ExplorerActivity* se stará o samotné vykreslení souborového systému a interakci s uživatelem.



Obrázek 15 *Diagram tříd balíčku converter.explorer*

Balíček *converter.epub_reader* obsahuje třídy pro zobrazení knihy ve formátu EPUB.

6.2 Oprávnění aplikace

Platforma Android obsahuje bezpečnostní opatření v podobě schvalování přístupu aplikace k systémovým či hardwarovým prostředkům. Metoda spočívá v tom, že všechny tyto operace musí být uvedeny v souboru *AndroidManifest.xml*, který obsahuje informace o aplikaci. Pokud by zde nebyla nějaká operace, kterou aplikace vyžaduje, skončí aplikace výjimkou. Tyto požadavky aplikace jsou zobrazeny uživateli při instalaci aplikace a musí je potvrdit. Pokud je nepotvrdí, instalace aplikace se přeruší. V souboru jsou také uvedeny hardwarové požadavky na zařízení a minimální verze API. Pokud telefon nepodporuje tuto minimální verzi, nepůjde aplikaci nainstalovat. Dále jsou zde definovány intent-filtry. Intent-filt definuje akci, kterou intent provádí a mimetype nesených dat. Jednotlivé filtry jsou přiřazovány aktivitám, které se starají o zpracování intentu.

V aplikaci budu využívat připojení k internetu. Dále čtení a zapisování na externí úložiště, abych mohl ukládat stažené stránky nebo je převádět do jiných formátů. Musím tedy tyto povolení uvést v souboru *AndroidManifest.xml*. A také potřebuji nadefinovat jeden intent-filter pro hlavní aktivitu, protože jedna z možností získání odkazu na stránku je sdílení stránky z prohlížeče.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
/* …*/
<intent-filter>
        <action android:name="android.intent.action.SEND" />
        <category android:name="android.intent.category.DEFAULT" />
        <data android:mimeType="text/plain" />
</intent-filter>
```

Výpis 2: Nutné povolení aplikace a intent-filter

6.3 Stažení stránky a uložení

Stažení stránky a uložení na externí úložiště je jedna z podstatných částí aplikace. Protože se jedná o úkol, který se skládá z menších částí, bude řešení úkolu rozděleno do více podkapitol.

6.3.1 Získání stránky z prohlížeče

Pro lepší komfort uživatele (aby nemusel stránku přímo psát do aplikace), jsem naimplementoval, způsob předání odkazu na stránku z prohlížeče. Uživatel tedy spustí prohlížeč, najde si stránku, kterou chce stáhnout a pomocí možnosti prohlížeče sdílet stránku a výběru aplikace Converter předá aplikaci požadovanou stránku.

Tato komunikace mezi aplikacemi se v systému Andorid provádí pomocí Intentu. Nejprve tedy zjistím, zda byla operace sdílení stránky vyžadována a potom (protože jsem nadefinoval intent-filter) zkontroluju samotný Intent. Pokud je vše v pořádku, předám text aplikaci a zobrazím.

Na výpisu 3 je část kódu, která kontroluje Intent. Výpis 4 potom ukazuje, jakým způsobem, jsou data vytažena z Intentu a zobrazena.

```
Intent intent = getIntent();
String action = intent.getAction();
String type = intent.getType();
//Check if Intent action is SEND and type not null
if (Intent.ACTION_SEND.equals(action) && type != null)
{
        //Check if type is plain text
        if ("text/plain".equals(type))
        {
                handleSendText(intent);
        }
}
```

Výpis 3: Kontrola intentu

```
private void handleSendText(Intent intent)
{
        String sharedText = intent.getStringExtra(Intent.EXTRA_TEXT);
        if (sharedText != null)
        {
                edit.setText(sharedText);
        }
}
```

Výpis 4: Metoda pro zobrazení odkazu

6.3.2 Stažení a pročištění stránky

Stažení stránky probíhá v samostatném vlákně. Andorid neumožňuje provádět stahování v hlavním vlákně, proto jsem vytvořil třídu, která dědí z třídy AsyncTask a ta vytvoří nové vlákno. Pro stahování stránky využívám knihovnu Jsoup, která stránku stáhne a vytvoří DOM.

Třída obsahuje metodu *doInBackground(String... uri)*, ve které probíhá stahování stránky statickou metodou parse třídy Jsoup. Metodě je předán odkaz na stránku a znaková sada souboru. Po úspěšném stažení je DOM uložen v objektu typu *Document* knihovny Jsoup. Na výpisu 5 je část kódu obstarávající stažení stránky.

Pro pročištění jsem vytvořil třídu, která obsahuje metodu *Parse()*. Metoda se stará o pročištění stránky. Nejprve v metodě převedeme DOM na text. Potom si nadefinuji tzv. whitelist, který obsahuje tagy, které chci po pročištění zachovat. Toto je vhodné pro eliminaci například škodlivého dynamického kódu. Následně zavolám statickou metodu clean třídy Jsoup a předám ji textovou podobu stránky a whitelist. Na výpisu 6 je podstatná část kódu metody, která se stará o pročištění stránky.

```
@Override
protected Boolean doInBackground(String... uri)
{
        Document document = null;
        try
        {
                document = Jsoup.parse(new URL(uri[0]).openStream(), "UTF-8", uri[0]);
                if(document != null)
                {
                        /* … */
                }
                /* … */
} catch (IOException e)
{
        e.printStackTrace();
        return false;
}
```

Výpis 5: Stažení stránky

```
public void Parse()
{
        /* … */
        String html = doc.toString(); // Html code
        Whitelist myTags = new Whitelist();
        myTags.addTags("body","div","h1","h2", "h3", "p", "a", "img", "b", "br", "strong",
                        "table", "td","tr", "ul", "li", "i", "header","article", "footer");
        myTags.addAttributes(":all", "class", ":all", "id", ":all","style", "img", "src", "a",
                        "href");
        String outputHtml = Jsoup.clean(html, myTags);
        /* … */
}
```

Výpis 6: Pročištění stránky

6.3.3 Uložení stránky

V aplikaci často potřebuji vytvořit nový soubor a uložit na externí úložiště. Ukládám také větší množství různých souborů. Proto jsem vytvořil abstraktní třídu, která se stará o zápis na externí úložiště a také kontroluje, zda je externí úložiště dostupné pro zápis. Ostatní třídy vytvářející nějaký soubor, dědí z této abstraktní třídy. Třída poskytující funkcionalitu uložení stránky se nazývá *HtmlFile*.

6.4 Stažení obrázků

Součástí webových stránek jsou i obrázky. Uživatel má tedy i na výběr stažení stránky s obrázky. V aplikaci tedy implementuju stažení jednotlivých obrázků stránky a přepsání cesty k obrázku. Na výpisu 7 je zobrazena kompletní metoda pro stažení obrázků.

Na výpisu 7 můžeme vidět metodu *getBitmapFromURL(String link)*. Metoda dostane v parametru odkaz na obrázek. Následně vytvořím připojení a stream, do kterého si načtu data obrázku. Potom dekóduju stream a uložím do objektu *Bitmap*. Tato metoda je volána v cyklu,

který projde všechny obrázky stránky a uloží je. Metodu pro ukládání obrázku nebudu z důvodu nezajímavosti ukazovat.

```
private Bitmap getBitmapFromURL(String link)
{
        try
        {
         URL url = new URL(link);
         HttpURLConnection connection = (HttpURLConnection)url.openConnection();
         connection.setDoInput(true);
         connection.connect();
         InputStream input = connection.getInputStream();
         Bitmap myBitmap = BitmapFactory.decodeStream(input);
         return myBitmap;
        }catch (IOException e){
                e.printStackTrace();
                return null;
        }
}
```

Výpis 7: Stažení obrázku

6.5 Konverze HTML stránek do EPUB formátu

V kapitole 2.2 jsem nastínil postup tvorby EPUB formátu z HTML stránek. První nutnou podmínkou je vytvoření adresářové struktury na externím úložišti, kde jedna složka bude určena pro ukládání stažených stránek. Tuto funkcionalitu poskytuje statická třída *Folders* z balíčku *converter.files_work*. Složky se vytvoří v případě, že ještě neexistují.

Dalším krokem při vytváření formátu je vytvoření souboru s deklarací MIME a container souboru. Vytváření těchto dvou souborů obstarává třída *Container*, která soubory vytvoří, pokud neexistují, protože jejich obsah se po celou dobu životnosti aplikace nemění.

Dalším souborem, který je nutné vytvořit je soubor s XML strukturou, názvem content a příponou opf. Soubor obsahuje informace o autorovi, které ponechávám vždy stejné. Důležitější částí souboru jsou samotné odkazy na jednotlivé stránky, které se budou vyskytovat v knize. Tyto údaje se dynamicky mění v závislosti na počtu stažených stránek. Nejprve tedy získám názvy všech stažených stránek a uložím je do seznamu. Jednotlivé odkazy musí být v elementu s názvem *item*, který je potomkem elementu *manifest*. V cyklu projdu seznam stránek a vytvořím jednotlivé elementy *item*. Všechny vytvořené elementy s daty zapíši do souboru a uložím do složky ke stránkám. O vytváření tohoto souboru se stará třída *Content*.

Posledním souborem, který je potřeba vytvořit, je soubor toc. Podobně jako soubor content se obsah mění dynamicky v závislosti na počtu stránek. Soubor určuje řazení jednotlivých kapitol. Rodičovský element pro jednotlivé kapitoly má název *navMap*. Element obsahuje dětské elementy *navPoint*, které obsahují data kapitoly. Element *navPoint* má dva dětské elementy. Jeden určující název kapitoly, který se bude zobrazovat v softwaru pro čtení a druhý obsahuje název stránky, která je v souboru content. Nejprve získám názvy všech stažených stránek a uložím do seznamu. Seznam stránek projdu v cyklu a pro každou položku

seznamu vytvořím strukturu potřebných elementů. Všechny vytvořené elementy zapíši do souboru a uložím do složky ke stránkám. O vytváření toho souboru se stará třída *Toc*.

Závěrečným krokem je vytvoření archívu adresářové struktury s příponou epub. Nejprve si vytvořím seznam, který naplním cestami k jednotlivým souborům. Potom vytvořím stream pro zazipování souborů. V cyklu projdu jednotlivé položky seznamu a zapíši obsah souboru do streamu. Funkci zazipování obsluhuje třída *ZipUtility*.

6.6 Konverze HTML stránek do FictionBook formátu

Oproti EPUB formátu FictionBook se skládá pouze z jednoho souboru, který je psaný XML syntaxí. Formát má definované XML schéma[2], soubor tedy musí být validní vůči schématu. Struktura souboru je rozdělena do tří hlavní částí. První částí je element *description*, který obsahuje další dětské elementy. Jsou zde uvedeny informace o knize a o autorovy. Další část souboru je obsažena v elementu *body*, který je rodičem elementu *section*. Elementů *section* může být více a obsahují samotný text knihy. Element *section* má další dětské elementy ve kterých je obsažen text knihy. V aplikaci vytvářím element *section* pro každou stránku nový. Poslední částí je uložení obrázků. Obrázky jsou vkládány na konec souboru do elementu *binary*, kdy každý obrázek je ve vlastním elementu. Jednotlivé obrázky v knize se odkazují na element *binary*, který nese data obrázku.

Pro tvorbu FictionBook formátu je tedy nutné projít všechny stažené stránky a získat z nich veškerý text a zapsat do souboru. Nejdříve tedy získám všechny stránky ze složky, která je určená pro uložení stažených stránek. V cyklu projdu postupně všechny soubory. Aktuální zpracovávaný HTML soubor si pomocí knihovny Jsoup převedu na DOM. Do seznamu si uložím všechny dětské elementy elementu *body*. V cyklu projdu všechny elementy a hledám takové, které obsahují text. Pokud najdu element obsahující text, získám jeho obsah a přidám data do *StringBuilderu*. Po průchodu všemi soubory zapíší data ze *StringBuileru* do souboru.

Na výpisu 8 je část metody, která provádí konverzi do FictionBook formátu. Z hlediska robustnosti metody jsem vynechal některé části pro úsporu místa. Část metody ukazuje nejprve převedení HTML stránky na DOM. Poté přidání do počátečních elementů do *StringBuilderu*. Dále získání všech elementů elementu *body*, následné procházení získaných elementů a kontrolu zda element obsahuje text. Pokud obsahuje text, probíhá kontrola, jestli se nejedná o speciální element např. o nadpis. V metodě kontroluju více speciálních elementů a různě je zpracovávám. Poslední věcí, která je obsažena v ukázce je přidání textu z elementu do *StringBuilderu*. Text získám pomocí metody *ownText()*.

Pokud nějaká ze stránek obsahuje obrázky, získám si cesty k jednotlivým obrázkům a uložím do seznamu. Po průchodu všemi stránkami a uložením textu do souboru projdu v cyklu seznam všech obrázků a postupně, každý zvlášť převedu do base64 a uložím na konec souboru v elementu *binary*.

[2] http://www.fictionbook.org/index.php/Eng:XML_Schema_Fictionbook_2.1

```
private void Convert()
{
        /* … */
        doc = Jsoup.parse(f,"UTF-8"); // create DOM
        data.addTag("<title>" + doc.title() + "</title>");
        data.addTag("<section>");
        List<Element> el = doc.select("body").select("*"); // body elements
        for(int i = 0; i < el.size(); i++)
        {
                Element e = el.get(i);
                if(e.hasText()) // element must have a text
                {
                        /* … */
                        if(e.tagName().equalsIgnoreCase("h1") ||
                          e.tagName().equalsIgnoreCase("h2") ||
                          e.tagName().equalsIgnoreCase("h3"))
                        {

                                data.addTag("<p><strong>" + e.ownText() +
                                                "</strong></p>"); // text in tag
                        }else{
                                data.addTag("<p>" + e.ownText() + "</p>"); // text in tag
                        }
                }
        /* … */
        }
        /* … */
}
```

Výpis 8: Část metody převodu do FictionBook

6.7 Otevření v nainstalovaných aplikacích

Výslednou knihu je možné otevřít z průzkumníka v mnoha různých aplikacích k tomu určeným. Implementoval jsem tedy funkčnost předání systému Android soubor. Systém sám nabídne z nainstalovaných aplikací tu, která podporuje předávaný formát. Tato operace se v systému Android provádí pomocí intentu. Na výpisu 9 je kód, který předá systému Android soubor. Důležité je nastavení akce intentu, která je ACTION_VIEW. Ta označuje, že chceme soubor zobrazit. Další důležitou věcí je předání intentu mimetype předávaného souboru. V této ukázce se jedná o "application/epub+zip" tedy o soubor ve formátu EPUB.

```
try
{
        Intent intent = new Intent();
        intent.setAction(android.content.Intent.ACTION_VIEW);
        File file = new File(adapter.getItem(pos).getPath());
        intent.setDataAndType(Uri.fromFile(file), "application/epub+zip");
        startActivity(intent);
}catch(Exception e)
{
  Toast.makeText(this, "Není žádná aplikace pro otevření", Toast.LENGTH_SHORT).show();
}
```

Výpis 9: Předání souboru systému Android

7 Testování

V této kapitole se budu zabývat testováním vytvořené aplikace. Při testování jsem odhalil několik chyb, které byly ve finální podobě aplikace opraveny. Jednou z chyb bylo při převodu do formátu FictionBook nedostatek paměti při vkládání obrázku. Chybu jsem odstranil snížením kvality obrázku na 70% a kompresí.

Testování probíhalo na dvou mobilních telefonech. První mobilní telefon byl Sony Xperia J. Tento telefon je vybaven jednojádrovým procesorem Qualcomm MSM7227A typu Cortex-A5 o taktu 1 GHz. Velikost operační paměti je 512 MB. Verze systému Android je 4.1.2 Jelly Bean. Druhé testovací zařízení byl mobilní telefon Sony Xperia P. Tento telefon je vybaven dvoujádrovým procesorem ST-Ericsson U8500 s jádry typu Cortex-A9 o taktu 1 GHz. Velikost operační paměti je 1024 MB. Verze systému Android je 4.1.2 Jelly Bean.

Testování na fyzických zařízeních bylo zaměřeno na rychlost konverze do jednotlivých formátů. Testoval jsem jak konverzi bez obrázků tak i konverzi s obrázky. V tabulce 2 jsou naměřené hodnoty pro mobilní telefon Sony Xperia J. V tabulce 3 jsou naměřené hodnoty pro mobilní telefon Sony Xperia P.

Tabulka 2 *Naměřené hodnoty testování na telefonu Sony Xperia J*

	EPUB	**FictionBook**	**Textový soubor**
Bez obrázků [s]	0.977	12.86	25.12
Počet stránek	70	70	70
Velikost souboru [MB]	0.5	0.5	0.3
S obrázky [s]	20.41	25.60	
Počet stránek / počet obrázků	70 / 680	70 / 680	
Velikost souboru [MB]	25.1	4.7	

Tabulka 3 *Naměřené hodnoty testování na telefonu Sony Xperia P*

	EPUB	**FictionBook**	**Textový soubor**
Bez obrázků [s]	0.931	9.81	17.89
Počet stránek	70	70	70
Velikost souboru [MB]	0.5	0.5	0.3
S obrázky [s]	13.48	18.18	
Počet stránek / počet obrázků	70 / 680	70 / 680	
Velikost souboru [MB]s	25.1	4.7	

7.1 Zhodnocení testování

Výsledky testování ukazují, že nejrychlejší konverze stránek bez obrázků je do formátu EPUB. Na obou zařízeních jsou časy konverze srovnatelné. Rychlost konverze oproti jiným formátům je dána tím, že stačí stažené stránky uložit do jednoho archívu. Převod do formátu FictionBook je značně pomalejší. Je to způsobeno tím, že každou stránku musíme otevřít a vybrat pouze elementy obsahující textové data. Nejpomalejší konverze je do formátu textového souboru, která je způsobena otevíráním jednotlivých stránek, vybráním textových elementů a nahrazením HTML entit tisknutelnými znaky.

Konverze stránek s obrázky byla rychlejší do formátu EPUB. Opět je to dáno tím, že stačí stránky spolu s obrázky uložit do jednoho archívu. Oproti tomu konverze do formátu FictionBook, která trvá delší dobu z důvodu vyhledání elementů obsahující textová data na stránce a konverze obrázků do base64. Z výsledků je patrné, že mobilní telefon s větším výkonem si poradí se složitější konverzí rychleji.

Závěr

Cílem bakalářské práce bylo navrhnout a naimplementovat aplikaci pro platformu Android, které bude provádět konverzi z webových stránek do vybraných formátů elektronické knihy a prostého textu. Dále nastudovat a vybrat vhodné formáty elektronické knihy, do kterých bude prováděna konverze a zjistit dostupnost čteček elektronických knih pro platformu Android.

Vytvořil jsem aplikaci, která dokáže stáhnout webovou stránku a provést konverzi do dvou formátů elektronické knihy a prostého textu. Aplikace dokáže stáhnout a zpracovat RSS kanál a umožňuje čtení jednotlivých článků nebo jejich stažení. Pro ty uživatele, kteří nebudou mít nainstalovanou některou z čteček elektronických knih, jsem naimplementovat vlastní jednoduchý zobrazovač výsledných knih. Pro komfort uživatele jsem naimplementoval průzkumníka, který dokáže otevřít knihu ve vlastním zobrazovači nebo knihu předat jiné nainstalované aplikaci v systému, která podporuje daný formát. Pro konverzi do jednotlivých formátů jsem naimplementoval vlastní algoritmus.

Ohledně budoucnosti aplikace a jejího dalšího možného vývoje určitě lze nalézt některé funkce, o které by se dala aplikace rozšířit. Jednou z těchto funkcí je přidání dalších formátů, do kterých bude realizována konverze. Další z funkcí je podpora více formátů elektronických knih vlastním zobrazovačem a přidáním některých dalších vymožeností, které zpříjemní čtení např. podpora denního a nočního schématu nebo ovládání jasu a mnoho dalších. Díky těmto vylepšením by uživatelé nemuseli mít externí aplikaci pro zobrazování knih. Dále by bylo možné přidat funkci, která bude při dostupném připojení k Internetu periodicky kontrolovat určitý RSS kanál a při nálezu nového článku upozorní uživatele.

Aplikace dosud není vystavena na obchodě Google Play. Publikace aplikace bude provedena až po rozšíření o některé z výše zmíněných funkcí, aby se zamezilo případným negativním ohlasům na aplikaci.

Použitá literatura

[1] MobileRead Wiki - eBooks. MobileRead [online]. 2014 [cit. 2014-04-08]. Dostupné z: http://wiki.mobileread.com/wiki/Ebooks

[2] EPUB | International Digital Publishing Forum. International Digital Publishing Forum | The Trade and Standards Organization for the Digital Publishing Industry [online]. 2014 [cit. 2014-04-08]. Dostupné z: http://idpf.org/epub

[3] MobileRead Wiki - ePub. MobileRead [online]. 2014 [cit. 2014-04-08]. Dostupné z: http://wiki.mobileread.com/wiki/EPUB

[4] MobileRead Wiki - FB2. MobileRead [online]. 2014 [cit. 2014-04-08]. Dostupné z: http://wiki.mobileread.com/wiki/FB2

[5] How to Create an EPUB File from HTML (and XML). Web Design - HTML XML - Web Development - Website Design[online]. 2014 [cit. 2014-03-28]. Dostupné z: http://webdesign.about.com/od/epub/a/build-an-epub.htm

[6] Jsoup Java HTML Parser, with best of DOM, CSS, and jquery. jsoup Java HTML Parser, with best of DOM, CSS, and jquery [online]. © 2009 - 2013 [cit. 2014-02-21]. Dostupné z: http://jsoup.org/

[7] SAX. SAX [online]. 2014 [cit. 2014-04-25]. Dostupné z: http://www.saxproject.org/

[8] AsyncTask | Android Developers. Wikipedia [online]. 13 Feb 2014 [cit. 2014-02-21]. Dostupné z: http://developer.android.com/reference/android/os/AsyncTask.html

[9] Activities | Android Developers [online]. 2014 [cit. 2014-02-27]. Dostupné z: http://developer.android.com/guide/components/activities.html

[10] Intent | Android Developers [online]. 2014 [cit. 2014-02-27]. Dostupné z: http://developer.android.com/reference/android/content/Intent.html

[11] Storage Options | Android Developers [online]. 2014 [cit. 2014-02-27]. Dostupné z: http://developer.android.com/guide/topics/data/data-storage.html

A Uživatelská příručka

A.1 Instalace a spuštění

Aplikaci lze nainstalovat a spustit na fyzickém zařízení (mobilní telefon, tablet), které obsahuje systém Android ve verzi 4.0 a vyšší. Pro spuštění lze také využít emulátor nainstalovaný na počítači.

A.1.1 Instalace na fyzické zařízení

Pro nainstalování aplikace jelikož se nenachází v obchodě Google play je nutné mobilní telefon propojit s počítačem pomocí USB kabelu. Jakmile je zařízení propojené jsou dvě možnosti jak aplikaci naistalovat.

1. přenést aplikaci na paměťovou kartu fyzického zařízení, kde aplikaci najdeme a nainstalujeme.
2. využít nástroj *adb* a příkaz: `adb [-s <sériové číslo>] install <cesta k aplikaci>`

V uvedeném příkazu část v hranatých závorkách, kde je, uvedeno sériové číslo je nepovinná. Je nutné v případě, kdy je k počítači připojeno více zařízení a my potřebujeme specifikovat na které zařízení se má aplikace nainstalovat. Příkaz `adb devices` slouží pro vypsání všech připojených zařízení spolu se sériovým číslem.

Pokud aplikaci instalujeme z paměťové karty je nutno ve fyzickém zařízení povolit instalaci z cizích zdrojů, protože aplikace nepochází z Google Play. (pokud bude instalace provedena přes nástroj *adb* není to nutné).

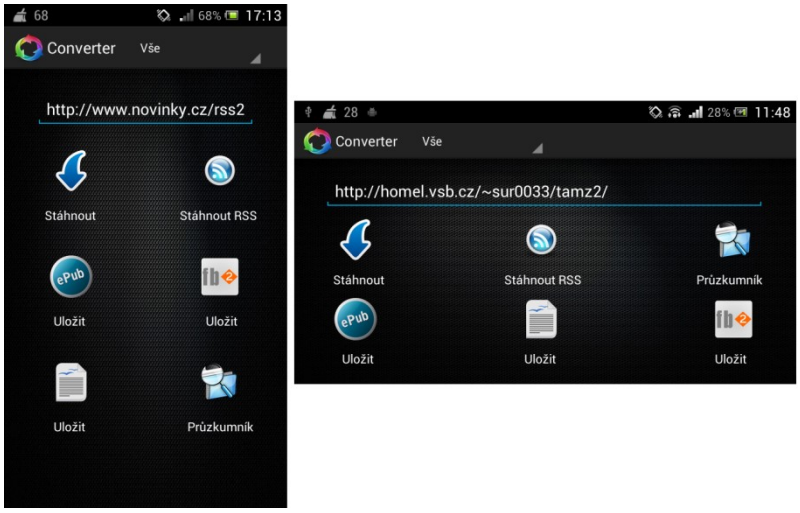
A.2 Práce s aplikací

Po dokončení instalace v reálném zařízení je možné aplikaci spustit. Jelikož pro práci s aplikací potřebujeme odkaz na požadovanou stránku, nejprve je možné si z prohlížeče předat odkaz na stránku, pomocí možnosti *sdílet s* v prohlížeči. Po spuštění aplikace se zobrazí hlavní menu (obrázek 16), ve kterém má uživatel možnost začít stahovat webovou stránku nebo stáhnout RSS kanál ze seznamu nachystaných kanálů. Po dokončení stahování jedné nebo více stránek může uživatel provést konverzi do formátů elektronické knihy nebo prostého textu. Další možností je stažení RSS kanálu nebo spuštění průzkumníku.

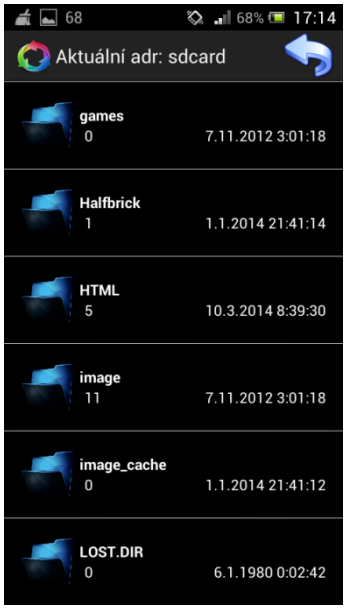
V průzkumníku (obrázek 17) má uživatel možnost si najít vytvořené knihy (HTML -> Knihy). Pokud uživatel přidrží prst na knize, se kterou chce pracovat, zobrazí se menu ve kterém má na výběr z možností zobrazení knihy ve vlastním zobrazovači nebo v jiné nainstalované aplikaci, která podporuje daný formát.

Ve zpracovaném RSS kanálu (obrázek 18) má uživatel možnost po kliknutí na titulek článku jej zobrazit. Pokud chce uživatel stáhnout jednotlivé články, použije zaškrtávací pole pro výběr článků a po dokončení výběru stiskne tlačítko menu a zobrazí se nabídka stahování.

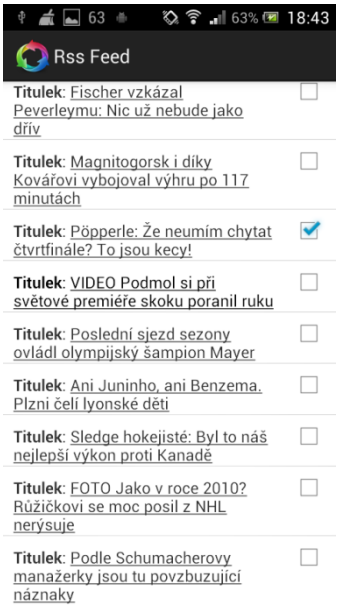
V zobrazovači výsledných knih má uživatel pro formáty EPUB a FictionBook možnost zobrazovat knihu po jednotlivých stránkách. Na obrázku 19 je zachycen vlastní zobrazovač výsledných knih.



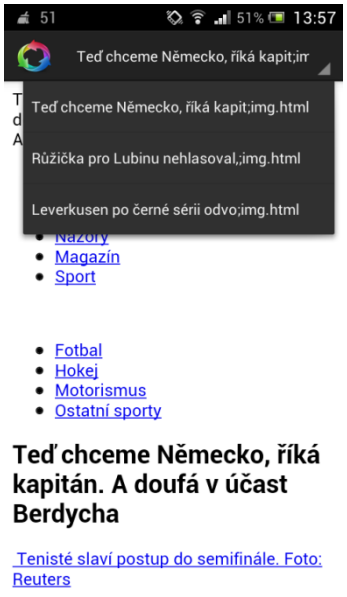
Obrázek 16 *Hlavní menu*



Obrázek 17 *Průzkumník*



Obrázek 18 *RSS kanál*



Obrázek 19 *Zobrazovač*

B Obsah elektronické přílohy

V elektronické příloze najdeme tyto adresáře:

- **Text** – text bakalářské práce ve formátech PDF a docx
- **Projekt** – zip archív celého projektu
- **Aplikace** – instalační balíček aplikace